# A Faceted approach to Service Specification

James Walkerdine, John Hutchinson, Pete Sawyer, Glen Dobson, Victor Onditi
*Computing Department*
*Lancaster University*
*Lancaster, LA1 4YR, UK*
*{walkerdi, hutchinj, sawyer, g.dobson, onditi}@comp.lancs.ac.uk*

## Abstract

*Service-centric computing is developing and maturing rapidly as a paradigm for developing distributed systems. In recent years there has been a rapid growth in the number and types of processes being proposed to support aspects of SOC. Many of these processes require that services be modelled in a particular way and this puts great pressure on traditional notions of service specification, questioning the very nature of how services should be described for potential consumers. We present a technique for addressing this theoretical and practical bottleneck: faceted service specification. This allows different specifications to exist side-by-side if they are needed, yet places little obligation on the service provider to support specifications that are judged to be of little or no value. We show how faceted service specification is being used in the SeCSE project to support advanced service-centric system development activities.*

## 1. Introduction

As Service Oriented computing (SOC) has matured, the experience gained has shed light on particular challenges. Part of this is the appreciation that the new paradigm does not make the inherently complex task of creating software systems in any way straightforward, instead it changes the nature of the problems faced. As such, the emergence and growing importance of services, particularly web services, and SOC are having a significant impact on software development.

In 2004, Leavitt cited a report predicting that worldwide spending on web service-based software projects would increase ten-fold in the five years to 2008, to around $11 billion, but reported a growing uncertainty amongst developers about supporting standards [10]. There is then a need for appropriate principles and practices to make best use of the available service technologies, from service identification and specification to service deployment [17]. However, to be adopted, they need to be pragmatic and have low overheads.

Software engineering experience has taught us that real applications require reliable processes for eliciting requirements, modelling business processes, architecture modelling, testing, establishing and measuring quality of service (QoS) and many other activities. When creating a service-based system, each activity requires information about the services being used or considered, since the consumption of services requires assessment and selection activities. This is a potential problem because, in general, there is only one information source, and that is its specification. Given the many different techniques available for carrying out these activities, the amount of information potentially required to fulfil every possible need is too great.

Within an SOC setting, many of these activities can be thought of as types of model-based reasoning. A potential consumer or user must reason about the offered service on the basis of a service model supplied by the service provider. It is our contention that the number of potential reasoning activities is so vast that no single model, or specification, would ever be sufficient. There is the very real potential for inconsistency in the information required for different purposes, so no single specification could be sufficient. The adoption of a "standard model" by the SOC community could mitigate these challenges. However, we would argue that no sufficiently rich model exists, and the required agreement is unlikely.

If services are to be more than just another implementation technology, and the creation of a genuine service marketplace is to be more than an optimistic dream, an adequate approach to service specification is needed. The approach must allow information needed about a service to be supplied as and when required without placing an unacceptable overhead on service providers and consumers.

In this paper, we describe the facet-based service specification approach adopted by the Service Centric System Engineering (SeCSE) project (IST 511680). Section 2 considers information requirements in the context of system development; section 3 describes service specification schemes and our faceted specification approach; section 4 discusses how faceted specifications have been implemented and provided with tool support. A real world case study is also provided.

## 2. Service-Centric System Development

The challenge in service specification comes from relating the information required about a service to the process of developing service-centric systems. As discussed above, part of the maturation process of SOC involves the proposal of processes for developing systems, and with them the challenges that must be addressed. Although specifications are *created* as part of service development, they are *used* during service-centric system development. Therefore, it is during the various processes of system development that the needs of specification are established. In this section, we highlight some of the processes being used in the SeCSE project and relate those to the information required to support them.

Service discovery is a simple term to describe a diverse set of processes that can be performed as part of system development. In particular, we differentiate requirements-based service discovery (RBSD) from architecture-based discovery and run-time discovery. Jones et al [7] describe how RBSD is an integral part of the requirements process. They point out that the

aim of the discovery process is not simply to identify services that match already specified requirements, but to assess the availability of services in order to determine the sort of system that can be developed. This is necessary if, for example, no appropriate services are identified, or if discovered services offer better solutions than those originally envisaged.

Early within the system development process, fine-grained technical descriptions are likely to be of little or no interest; instead what is required is a description of the service in a form that is intelligible to humans. Developers can then carry out a first pass filter of the available services according to the needs of the system being developed, with only partial requirements to work with. This approach contrasts with those that assume that services are matched to a set of specified requirements using ontologies (e.g. OWL-S [4]) or category based searching (e.g. UDDI [21]), in that it recognizes the value of cross-domain fertilization and the relationship between service discovery and requirements discovery.

Further forms of filtering can also help to determine the form of the system being developed and to identify services that may be used. The information used to make these judgments can include management/commercial information such as usage costs and information about service level agreements (SLAs), or much more detailed technical information about the service, such as operational semantics. Consideration of a service's operational semantics does not serve only to filter candidate services. Along with information about exceptions, it acts to constrain the design of the system being developed. However, the architectural considerations of the system may also be used to further identify suitable services. This places something of a burden on the way a service's operational semantics are specified.

For example, within SeCSE, BPEL or OCL may be used for specifying operational semantics. In part, this is a reflection of the differing needs the specifications are intended to support and, in part, it is a reflection of the different attitudes to the trade-off between completeness and ease of use. What is clear is that there is currently no single specification scheme that addresses all of the necessary issues.

A similarly complex area of service specification concerns QoS because it relates to a number of different areas of the service-centric system development process. It is also likely that the QoS that a service can provide, and guarantee, will become an even more important distinguishing factor as more services become available. QoS issues may be a factor in the coarse-grained filtering, the more detailed filtering of candidate services, in the selection of services in architecture-based or run-time discovery, or more generally when a service consumer has specific performance requirements and desires that these to be bound in an SLA with a service provider. This potentially broad applicability of QoS concerns makes adequate treatment of QoS specification an important challenge.

The challenge is compounded by the lack of standards for describing QoS attributes, with the result that there is no guarantee that providers will express their services' QoS in the same way that users express their QoS requirements. Yet the representations need to be compatible, not only syntactically but also semantically – or there must exist some means of translating between the two – if QoS specifications are to be used successfully as part of the service-centric development process.

For example, a user requires a response time of < 0.5 seconds. A provider states that their service has an average response time of 250 milli-seconds. Common sense would suggest that the service was potentially compatible with the user's requirements and should be considered further, but this requires us to translate units to come to this conclusion.

Within SeCSE, we have a QoS ontology available that focuses on certain dependability aspects such as availability and reliability [5], and are using this in conjunction with an ontology based specification as a means of addressing some of these issues. It is apparent, however, that in such an immature field, it is likely that other ways of representing QoS will be proposed and used because of the potentially wide applicability of QoS constraints.

## 3. Specification Schemes and Facets

Providing a specification for a service is a means of supplying potential users with a model of its features and/or behaviour so that the potential user can assess its appropriateness and, when it is used, predict its behaviour. As such, the specification is not a complete representation of the service being offered, but a projection that is intended to supply the potential user with the information they require to reason about, and consume, the service.

In most real-world scenarios, this property of specification (i.e. the lack of completeness of the representation [14]) is its very *raison d'être*: it hides information that is deemed unimportant to potential users and allows them to concentrate on what is important to them. Consider, for example, the many different types of representation used in the Unified Modeling Language (UML). Each is designed to model aspects of the system according to a particular point of view.

We saw above some of the different types of information required by potential users/developers to assess and use services, each for a different purpose. Table 1 lists some of the many technologies proposed for service specification. In producing a mechanism for managing service specification, we are interested in covering the aspects addressed in Table 1.

The table highlights the large number of different, often competing, mechanisms that can be used to specify each aspect. In some cases the capabilities they provide overlap, but for others the technology is unique in that it captures a certain type of information that others do not. Obviously it is not realistic to expect that service providers will support all of the technologies/schemes proposed, and to ease this some can already be used alongside each other (for example, UDDI and OWL-S can be integrated with WSDL). However, the case still remains, that in order to produce a comprehensive and useable specification, a variety of specification technologies will typically need to be drawn upon.

Rather than attempting to define a single, all-encompassing specification scheme, we use a set of *facets*, whose primary purpose is to bring together, and bring order to, specifications that are expressed in different schemes or languages that address similar properties of a service. Each facet focuses on one or more service properties (e.g. general description, binding, etc).

| Specification Type | Technology | Comments |
|---|---|---|
| Service Description | Text | Freeform, thus scope unlimited. Easy to use. Hard to interpret |
| | UDDI | UDDI supports the provision of structured textual descriptions. A mature standard. |
| | OWL-S | OWL-S uses structured textual descriptions and ontologies. Supports the creation of a semantic description of a service |
| | WSMO [25] | A less mature alternative to OWL-S for capturing semantic descriptions |
| | UML [9] | Support for component based descriptions of services |
| Service Signature | Text | Freeform, thus scope unlimited. Easy to use. Hard to interpret |
| | WSDL [23] | The standard technology for specifying the syntax of a services interface. A mature standard. |
| | UML | Supports the modelling of interfaces and ports |
| | WSDL-S [13] | An extension to WSDL that supports semantic descriptions of the operations |
| Operation Semantics | Text | Freeform, thus scope unlimited. Easy to use. Hard to interpret |
| | OWL-S | Supports the specification of pre-/post- conditions and effects |
| | UML/OCL | UML can incorporate OCL that can be used to specify pre-/post- conditions for services |
| | WSMO | Supports the specification of pre-/post- conditions and effects |
| | WSDL-S | Supports the specification of pre- and post- conditions for operations |
| Behavioural specification | Text | Freeform, thus scope unlimited. Easy to use. Hard to interpret |
| | UML | Supports the modelling of states and of state transitions |
| | OWL-S | Supports state modelling and process modelling |
| | OpenModel [6] | Supports comprehensive behavioural modelling. Not widely used |
| | BPEL4WS [3] | Supports business process modelling |
| | WSCI [26] | Supports the modelling of communication between services |
| Quality of Service (currently there exists no standard for specifying the QoS attributes of a web service) | Text | Freeform, thus scope unlimited. Easy to use. Hard to interpret |
| | UDDIe [22] | } - Provides QoS extensions for use with UDDI |
| | SWSQL [1] | } |
| | WSOL [20] | } |
| | E QoS [16] | } |
| | Extensible QoS Model [11] | } - Provides QoS extensions for use with WSDL |
| | WS-QoS [19] | } |
| | WSLA [12] | Supports the specification and monitoring of QoS with the use of electronic SLA's |
| | WSML [24] | Supports the expressing of SLA's |

Table 1. Specification schemes in SOC

For example, an Operational Semantics facet may embed OCL or BPEL based specifications, or both if desired, that describe service behaviour. By also supporting the use of third party specification mechanisms, the faceted service specification model can maintain compatibility with other approaches, and both current and future developments.

Facets bear some resemblance to viewpoints as used in requirements engineering (e.g. [8]). Although the resemblance is valid, they serve entirely different purposes. Viewpoints are a projection over requirements, used to elicit and structure them according to different points of view. Facets are a projection over service functionality, used to achieve specification for a purpose. There are also similarities between facets and the viewpoint proposed in the Open Distributed Processing reference model (RM-ODP) [15]. Although proposed for use in component-based development [18], RM-ODP offers a fairly rigid framework that aims for completeness in its domain and cross-viewpoint consistency [2], rather than the flexible incorporation of developing standards offered by facets.

The faceted specification approach addresses some key challenges:

- The ability of service consumers to discover suitable services is enhanced by facets and specifications, and the languages used to express them, being represented explicitly. Consumers can tell immediately if the information needed to evaluate a service against their requirements is available, and if it is in a form that is intelligible to them.
- Prescriptivism is avoided by allowing service providers to produce only the specifications they choose to use to support their intended customers.
- Flexibility is ensured because providers can add new facets and new specifications, if doing so adds value. If a specification type is superseded by a new language or scheme, not only can the new scheme be included, but the obsolete scheme or language can be omitted. There is no obligation to maintain specifications, or facets, if they are no longer used.

## 4. Implementing and Supporting Faceted Specification

An early activity in the SeCSE project has been to define a conceptual model for services and their context, and this is essentially a prerequisite for defining an initial set of facets to support. The conceptual model acts as a device for clarifying understanding, and whilst having "supported facet types" may seem counter to the aim of extensibility, they are intended to support the added-value processes being developed as part of the project (c.f. Section 2). As such, the facets types listed below are not intended to be complete, merely to address the needs of project partners. Currently, then, we support the following facet types:

**Signature**: this provides the same information as a WSDL specification. WSDL will normally be the language used to specify this facet.

**Description**: this supports RBSD. A structured natural language description is used to support semantic matching.

**Operational Semantics**: augments the signature specification of service operations with information permitting the semantics of operations to be understood and evaluated by service consumers.

**Exception**: permits service failure behaviour to be described along with associated pre-/post- conditions.

**QoS**: this permits service QoS to be described. QoS denotes a range of non-functional properties and understanding of it in the service engineering context is developing rapidly (e.g. which metrics to use and the role of SLAs).

**Commerce**: intended to make commercial information available for service consumers. This can include information about the Service Provider, cost of using the service, SLA's, and additional information such as policies adopted by the provider/service.

**Testing**: this makes test cases and test data available to service consumers.

**Management**: intended to capture management information related to the service specification, in particular versioning information, change history and information to do with the type and granularity of the service specification.

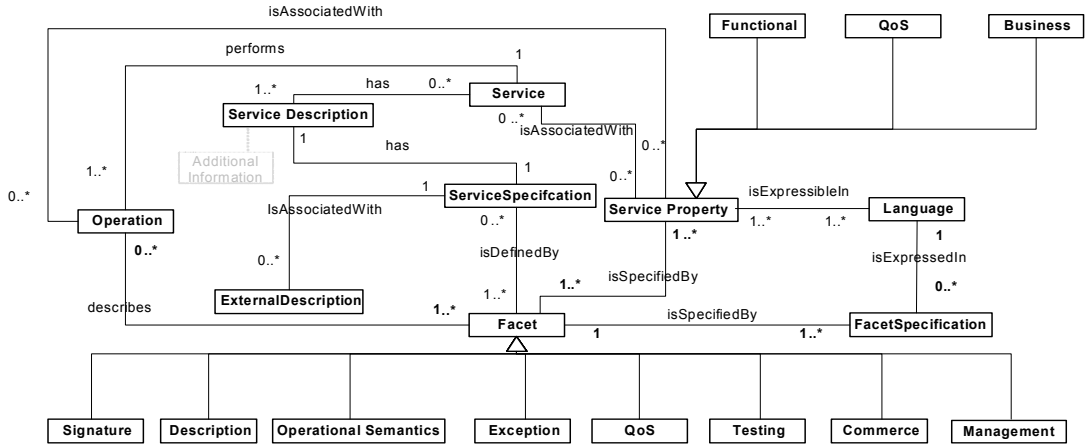


Figure 1. Specification Conceptual Model

As can be seen, in addition to information about operations that is conventionally considered part of a service's specification, facets are intended as a general mechanism for publishing information about a service (e.g. QoS and management information).

Figure 1 shows the service specification part of the conceptual model. One key element is that service properties (function, QoS and business) are specified by facets, expressed in *language specific facet specifications*. Although the facet specifications above describe the intended focus of the facet types, the conceptual model does not limit the types of service properties that may be specified in a given facet. This apparent breach of the principle of separation of concerns is not only justifiable, it is desirable. A specification scheme may primarily support reasoning about particular types of service properties. However, it may contain any information that is deemed necessary to support this reasoning. (For example, the commerce facet provides information about cost, but to do so adequately, it may include basic information about operations and QoS.) Therefore, to ensure the flexibility that is an advantage of the faceted approach, we cannot restrict the types of service properties that may be described in a given facet type. However, they do have an intended focus.

Logically, faceted service specifications are based on a three-tier hierarchy: a service specification references zero or more facets which in turn reference one or more language specific specifications. This is actually implemented as a two-tier file structure. The top level "service specification" includes the details of each of the facets supported by that specification. Each facet then references the language specific specifications that it supports. There are a number of reasons for this. From a practical point of view, there is a trade off between the size of the specification files and the number of accesses required to gain the specification. A single specification file, including all of the available specifications could potentially be very large, and is likely to include information not required by many potential users – this is part of the faceted specification philosophy. However, if the logical three-tier structure were carried over into a file structure, potential users would be forced to make multiple accesses to find out what information was available. By implementing a two-tier file structure, we support one of the key advantages of faceted specification – explicitness – by making the available specifications, and the languages used available in the top-level file. Users can then access the individual specifications they require, and not those they have no use for. Figure 2 shows the file structure for the two levels of files used to implement facets. Figure 2 (a) shows the top-level file structure, and Figure 2 (b) shows the language specific specification file structure.

The notion that new types of facet and facet specifications can be added raises the question of how a service consumer can ascertain whether or not the available specifications are intelligible to him/her. Although we define a set of specification structures within SeCSE, the flexibility of the faceted approach means that a service developer/provider is also able to provide their own. In order to assist a service consumer in exploiting such specifications, the faceted approach also encourages the development of schema models for the specifications used. These schemas are then stored alongside the specification within the registry allowing them to be accessed and assessed by consumers during the discovery process. Obviously, this does not address issues related to specification semantics, however there is not yet a pragmatic and feasible mechanism for tackling this in an automated fashion and we assume that human involvement will be required.

The introduction of new facets is more straightforward as these are more closely embedded within the faceted approach, as illustrated in the next section.

```
<ServiceSpecification>
    <ServiceName> </ServiceName>
    <ServiceID> </ServiceID>
    <ServiceSpecificationLastEdited> </ServiceSpecificationLastEdited>
    <ExternalSpecificationLink> </ExternalSpecificationLink>
    <Facet>
        <FacetType> </FacetType>
        <FacetOwner> </FacetOwner>
        <FacetSpecification >
            <ReferencedOntology> </ReferencedOntology>
            <ReferencedSIM> </ReferenceSIM>
            <FacetSpecificationLanguage> </FacetSpecificationLanguage>
            <FacetSpecificationLink> </FacetSpecificationLink>
        </FacetSpecification>
    </Facet>
</ServiceSpecification>
```

(a) Top level specification, including facets

```
<LanguageSpecificSpecification>
    <FacetType> </FacetType>
    <ReferencedOntology> </ReferencedOntology>
    <ReferencedSIM> </ReferencedSIM>
    <FacetSpecificationLanguage> </FacetSpecificationLanguage>
    <FacetSpecificationOwner> </FacetSpecificationOwner>
    <FacetSpecificationLastEdited> </FacetSpecificationLastEdited>
    <FacetSpecificationData> </FacetSpecificationData>
</LanguageSpecificSpecification>
```

(b) Language specific specification

Figure 2. Two tier specification file structure.

### A. *Facet Management Tool*

Support for our faceted service specification approach is provided by the Facet Management Tool, which can be used by service providers to help create, specify and manage the facets within a service specification. Figure 3 shows the Facet Management Tool in use. It shows a faceted specification for the "Business Trip" service. In the top right a table displays the facets that exist within the specification, below this is a preview pane that can be used to view the specifications within a selected facet. The tool currently supports:

*Facet Management* - the creation and editing of facets. A default set of facets can also be defined for service specifications.
*Specification Management* - allows the importing of specifications into facets. The tool can handle Natural language specifications and those that are derived from XML. This includes XMI based UML.
*Editor Management* - allows the assigning of 3rd party editors to different specification types, which can then be launched directly from the tool. This helps support the flexibility of the faceted approach.
*Facet Forms* - support the use specification guidance forms for individual facets. These forms represent a 'set specification structure' for the facet and in particular allow for better integration with the tools being developed by SeCSE partners.
*Consistency mechanisms* - mechanisms have been developed for checking consistency across specifications. These ensure that, for example, operation signatures are consistent throughout.
*Facet File Generation* - once the user has built up a faceted specification, the tool can automatically generate facet specification files in XML (XML Schema is used for the schemas). These files can then be incorporated within a SeCSE (or SeCSE-compliant) registry.

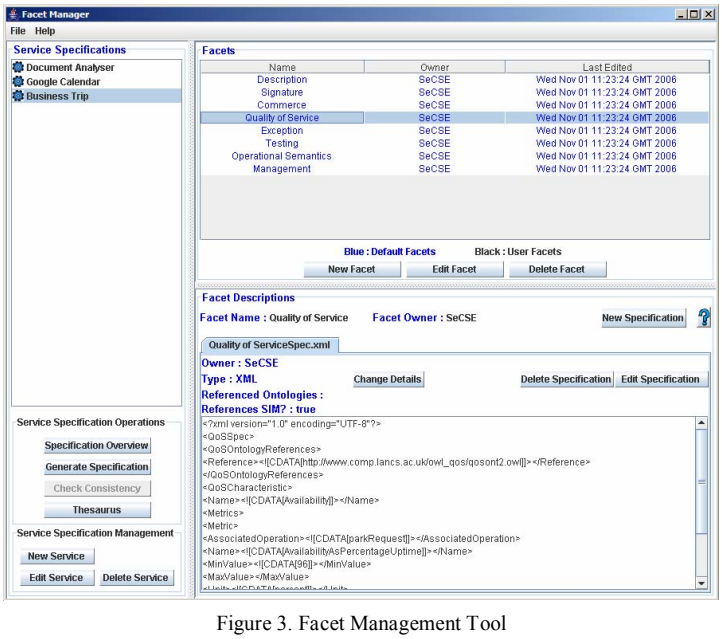


Figure 3. Facet Management Tool

### B. *Case Study: The Business Trip Service*

The Business Trip service is being developed within SeCSE by the Fiat Research Centre (CRF) as a service that can be utilised by drivers in their future range of cars. At present the service offers two capabilities:

- *A Journey Scheduler* - that allows a journey to be planned and monitored (based on location and time).
- *A Car Parking Booker* - that lists nearby car parks that possess free spaces, and allows the driver to book a space.

Alongside this development, a corresponding faceted service specification was also built by CRF. The specification consisted of Description, Signature, Commerce, QoS, Operational Semantics and Management facets, with each possessing a facet specification (c.f. figure 3).

The Description facet specification utilised a SeCSE specification structure that is specifically geared to support the process of RBSD. Figure 4a depicts the structure of this specification, and figure 4b shows it being used to specify the Business Trip service within the context of the tool. A similar approach was used for the other facet specifications.

```
<FacetSpecificationData>
    <Description>
        <ServiceGoal/> - the goal of the service
        <ServiceConsumers> - expected consumers of the service
            <Consumer/>
        </ServiceConsumers>
        <ShortServiceDescription/> - brief description of the service
        <LongServiceDescription> - longer description of the service,
                    including a breakdown of the operations it offers
            <ServiceOperations> - a list of the services' operations
                <Operation/>
            </ServiceOperations>
        </LongServiceDescription>
        <ServiceRationale/> - the rationale behind the service
        <Assumptions> - any assumptions the service has
            <Business/>
            <Technical/>
        </Assumptions>
        <PreConditions/> - the pre-conditions for this service
        <PostConditions/> - the post-conditions for this service
        <Miscellaneous/> - any additional miscellaneous information
                                (for example, contextual information)
    </Description>
</FacetSpecificationData>
```

Figure 4a. Structure for the SeCSE Description specification

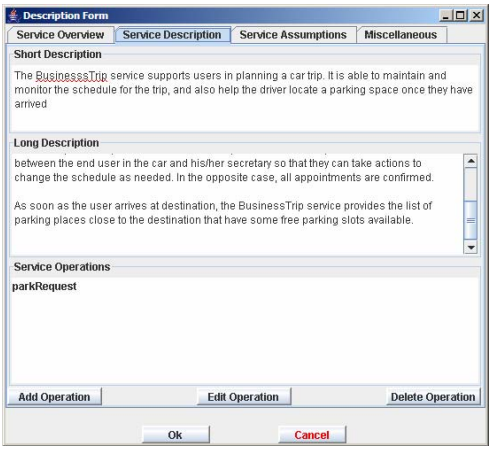


Figure 4b. Corresponding facet form

After the faceted specification was created it was deployed within a registry where it, in turn, became part of a RBSD activity to assess the specifications usefulness. Initially a natural language based discovery was performed utilising the Description facet specification as shown above. The RBSD techniques sifted through 100 similar specifications and produced a shortlist of four - one being the Business Trip faceted specification. The second stage of the RBSD then focused on refining this shortlist against a set of QoS criteria. As part of this process the QoS ontology was utilised to allow for the translation of the different time metrics used within the query and the QoS specification. The Business Trip service was found to be the most suitable candidate.

### C. *Current Status and Future Work*

Development of the faceted specification approach is still ongoing, with facets and tool support still being finalised. Consequently, a comprehensive evaluation of the specification approach is still to be performed. However, initial evaluation has taken place and, over the last year, the approach has been successfully used by industrial partners to create over 100 specifications for services from the telecommunication and automotive industries. The specifications produced have been those required to enable the SeCSE service-centric system development processes.

Feedback from the partners has largely been positive, with it being found to be sophisticated enough to allow for detailed specifications, but on the other hand also flexible enough to cope with the different specification practices that can exist within organisations.

The modular and extensible nature of the faceted specification approach was well received. Developers found that it helped them to organise their specification, whilst still being understandable and usable. Likewise, they liked the fact that new facets could be defined and facets themselves were not restrictive in the type of specification language they could accommodate.

During the next cycle of the SeCSE project a more comprehensive evaluation of the faceted approach will be performed, which will also involve it being used alongside other SeCSE developments.

## 5. Conclusions

The growing maturity of SOC is bringing a greater awareness of the problems that must be addressed if service-based systems are to be successfully implemented and adopted. With that comes the need to reason about services in a number of different ways, using the service specification as the only information resource. We have presented a faceted approach to service specification, which seeks to provide an extensible structure for managing the different formalisms that can be used to describe services, whilst ensuring that service providers have the greatest degree of flexibility possible for utilising emerging technologies for describing their services for their target audience.

The faceted service specification approach presented has been developed as part of the SeCSE project in order to support the added-value techniques and tools being developed by project partners. The types of specification that these techniques rely on vary greatly and lend weight to the view that there is not such thing as a "one size fits all" approach to service specification. The plethora of competing specification schemes and technologies also support this belief. The faceted approach allows multiple specifications to be supported, if their inclusion makes a particular type of reasoning about services possible. However, it avoids prescribing particular specification schemes, and instead allows providers to choose what information they make available.

The approach is supported by a Facet Management Tool and is currently being used by project partners. Early evaluation confirms the efficacy of the approach.


## ACKNOWLEDGEMENTS.

We would like to thank all of our partners in the SeCSE project consortium. The work is funded under EU Integrated Project 511680.


## REFERENCES


[1] Bilgin, A., Singh, M. (2004) "A DAML-Based Repository for QoS-Aware Semantic Web Service Selection", Proc. IEEE Int'l Conf on Web Services (ICWS'04), San Diego, USA.

[2] Bowman, H, Derrick, J, Linington, P, Boiten, E and Steen, M, (1996), Cross Viewpoint Consistency in Open Distributed Processing, Software Engineering Journal, 11(1):44-57, 1996.

[3] BPEL4WS Curbera, F., et al. 2002. Business process execution language for web services. http://www-106.ibm.com/developerworks/webservices/library/wsbpel/

[4] DAML-S (2004) OWL-S 1.1. http://www.daml.org/services/owl-s/1.1/

[5] Dobson, G., Lock, R. and Sommerville, I., "QoSOnt: a QoS Ontology for Service-Centric Systems" in Proceedings of Euromicro SEAA, pp. 80-87, 2005

[6] Hall, R., Zisman, A. (2004) "Behavioural Models as Service Descriptions", Proc. ISOC'04. New York, USA.

[7] Jones S.V., Maiden N.A.M., Zachos K. & Zhu X., 2005, 'How Service-Centric Systems Change the Requirements Process', Proceedings REFSQ'2005 Workshop, in conjunction with CaiSE'2005, 13-14 2005, Porto, Portugal, 105-119.

[8] Kotonya, G and Sommerville, I, (1997). Requirements Engineering — Processes & Techniques. John Wiley & Sons.

[9] Kruger, I. H., (2002) "Towards Precise Service Specification with UML and UML-RT", Proc. Workshop on Critical Systems development with UML, Dresden, Germany. http://www4.in.tum.de/~csduml02/27.pdf.

[10] Leavitt, N., "Are Web Services Finally Ready to Deliver?", IEEE Computer, 37(11), 14-18.

[11] Liu, Y., Ngu, A., Zeng, L. (2004) "QoS Computation and Policing in Dynamic Web Services" Proc. WWW2004, New York, New York, USA.

[12] Ludwig, H., Keller, A., Dan, A., King, R., Franck, R. (2003) "Web Service Level Agreement (WSLA) Language Specification". http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[13] Miller, J., Verma, K., Rajasekaran, P., Sheth, A., Aggarwal, R., Sivashanmugam, K. (2004) WSDL-S: A Proposal to W3C WSDL 2.0 Committee. http://lsdis.cs.uga.edu/projects/WSDL-S/wsdl-s.pdf

[14] Rechtin, E and Maier, M. (eds), "The Art of Systems Architecting (Systems Engineering)", CRC Press, 1996.

[15] RM-ODP , International Standard Organisation (ISO), Information technology - Open Distributed Processing - Reference model: Overview, ISO/IEC JTC1/SC07, 10746-1, ITU-T Recommendations X.901 (1996).

[16] Shen, D., Yu, G., Nie, T., Li, R., Yang, X. (2004) "Modeling QoS for Semantic Equivalent Web Services". Proc. Fifth International Conference on Web-Age Information Management (WAIM 2004), Dalian, China.

[17] Stojanovic, Z., Dahanayake, A. and Sol, H.: "A Service-Oriented Component Modeling Approach". Information Modeling Methods and Methodologies, 2005: 300-322.

[18] Stojanovic Z., Dahanayake, A.N.W. Sol, H.G., (2001), Integration of Component-Based Concepts and RM-ODP Viewpoints, the 1st Workshop on Open Distributed Processing WOODPECKER 2001, Setubal, Portugal, July 6-7, 2001, pp. 98-109.

[19] Tian, M., Gramm, A., Naumowicz, T., Ritter, H., Schiller, J. (2003) "A Concept for QoS Integration in Web Services" 1st Web Services Quality Workshop (WQW 2003), Rome, Italy. www.wsqos.net

[20] Tosic, V., Patel, K., Pagurek, B., (2002) "WSOL - Web Service Offerings Language", Proc Workshop on Web Services, e-Business, and the Semantic Web (WES) Toronto, Canada, pp. 57-67.

[21] UDDI Technical White Paper, September 2000. http://www.uddi.org

[22] UDDIe project. http://www.wesc.ac.uk/projects/uddie/uddie/

[23] W3C. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl

[24] Web Services Modelling Language. http://www.wsmo.org/wsml/.

[25] Web Service Modelling Ontology. http://www.wsmo.org/.

[26] WSCI Arkin, A., et al. 2002. Web services choreography interface WSCI. http://www.w3.org/TR/wsci